

MARCH 1967/MTP-57

on-line text processing introduction and overview

D. E. WALKER

Information Sciences Department

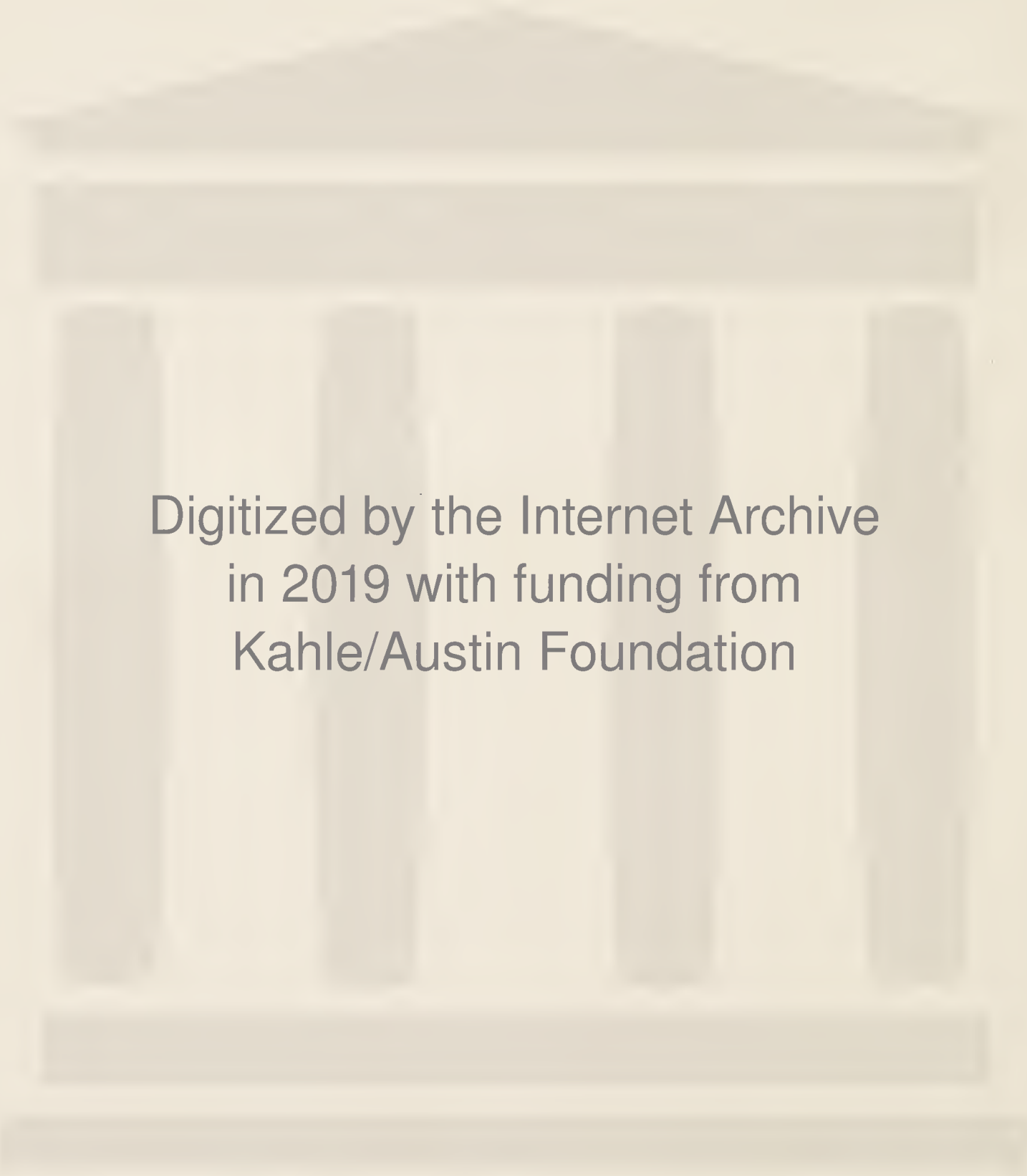
The MITRE Corporation

ON-LINE TEXT PROCESSING:
INTRODUCTION AND OVERVIEW

by

D. E. Walker

March 1967



Digitized by the Internet Archive
in 2019 with funding from
Kahle/Austin Foundation

<https://archive.org/details/onlinetestproces57dwwa>

Acknowledgments

The group responsible for the design of the text-processing system and the programming of its initial implementation included Renée Beller, Carter Browne, Paul Chapin, Stanley Cohen, Jeanne Fleming, Richard Glantz, Louis Gross, Ted Haines, Lewis Norton, and David Vetter.

TABLE OF CONTENTS

	Page
I. OBJECTIVE	1
II. APPROACH	3
A. Introduction	3
B. Linguistic and Logical Framework	4
C. List-Processing Programming Languages	6
D. An On-Line Computer Facility	8
III. CURRENT STATUS	10
A. Introduction	10
B. Preliminary Model	10
C. System Design	15
D. Proposed Additional System Components	17
1. Transformational Analysis	17
2. Morphological Analysis	19
3. Thesaurus and Dictionary Procedures	20
4. Semantic Analysis	21
5. Discourse Analysis	22
E. Computer Facility	22
F. The TREET Programming System	23
G. Program Structure for the Text-Processing System	25
REFERENCES	26

I. Objective

The objective of the current work program is to develop an on-line computer capability that would assist an information analyst in processing textual materials. The high volume, uncontrolled format, and high topical density found in many kinds of textual data make mechanical aids extremely desirable, but these aids can be effective only to the degree that they support the analyst's distinctively human inferential and evaluational skills. Although the computer capability certainly must provide the analyst with a coherent and structured basis in which the topical relevance of material is directly expressible and recoverable, at the same time it must be flexible enough to enable him to test hypotheses by organizing material in alternative ways.

The initial result of the work program is the design of a system for text processing and a computer implementation of a preliminary model of the system that strongly supports the validity of the basic design concepts. The model system allows an analyst to scan through messages, reports, and other documents and to identify significant content structures which are processed linguistically and then stored in the computer in a logical form. With respect to a given area of interest the analyst is able to search through the stored content representations, identify those that seem relevant, recover the text passages from which they were derived, and edit and annotate these passages to generate reports.

The preliminary model of the text-processing system is operational on an IBM 7030 computer with DD-13 display consoles. Work in progress is directed toward the implementation of an augmented system in an IBM System/360 facility. Initially, the augmented system is expected to be used as an experimental vehicle by information analysts to structure their own personal files. Their experience with the system should establish whether the basic logical and linguistic concepts could prove suitable for structuring files that would be accessed by a number of analysts. Longer range goals include an evaluation of the suitability of a system of this kind for organizing a document library. If a framework for representing information content could be shared by analysts and librarians, it could lead to an integrated system for handling textual data. At the very least, such a shared framework would encourage more effective use by each group of the work of the other.

II. Approach

A. Introduction

The design of the text-processing system has two major characteristics: first, linguistic analysis and logical formalization are basic procedures; second, man/machine interaction is essential.

The stress on linguistic analysis and logical formalization is based on a conviction that the progressive mechanization of textual information content can be accomplished most effectively through such techniques. The development of transformational grammar, in particular, has provided a formal, mathematical model of language and a rigorous basis for computer programs that derive the syntactic structure of a sentence mechanically. Concomitantly, recent non-standard elaborations of the functional calculus have resulted in the use of complex predicate-argument structures as normalized representations of textual information content. Furthermore, current research within transformational theory in both syntax and semantics is leading to a characterization of the base or underlying structure of a sentence which is directly interpretable in terms of such predicate-argument relationships. As a result there is a convergence of the results of linguistic research and the results of work in applied logic that is particularly relevant for text processing.

Interaction between man and machine is essential in a text-processing system, because the depth of analysis necessary for representing the information content of text through linguistic and logical techniques cannot yet be wholly mechanized. In addition, although a computer can provide capacity and speed, the analyst is needed to monitor and guide its operations at critical points in the flow of processing. The design developed

for the project allows the responsibility for providing the mechanics of analysis to be shifted progressively from the analyst to the computer as more sophisticated procedures become available.

Within the system under development the two major characteristics of the design are reflected directly both in the programming structure and in the kind of computer facility used. Both linguistics and logic involve complex non-numerical data structures. List-processing programming languages have proved to be particularly well suited to the manipulation of these kinds of structures. List-processing languages have also been used extensively on-line in time-shared computer systems to facilitate communication between the user and the computer. In the text-processing system the requirements for man/machine interaction make on-line control essential.

B. Linguistic and Logical Framework

The theory of linguistic structures underlying the project work effort is that of transformational generative grammar as developed by Chomsky and his associates at M.I.T. and elsewhere, (e.g. Chomsky, 1965; Katz and Postal, 1964). The theory postulates three components of a grammar: syntactic, semantic, and phonological. Within syntax, phrase structure rules generate base trees representing the underlying structure of a sentence. Semantic rules interpret these base structures to yield the meaning of the sentence. The transformational rules of the syntactic component convert base trees to surface trees. Phonological rules operate interpretively on the surface trees, resulting in the sequence of words in the sentence as it would be pronounced.

The early work on transformational grammar at the MITRE Corporation resulted in two major achievements: (1) the preparation of the first large coherent transformational grammar of English in computer-compatible form, and (2) the development and programming of an analysis procedure in terms of that grammar which produces base trees for a significant class of English sentences (cf. Walker, 1965; Zwicky, et al., 1965; Walker, et al., 1966).

The project work effort has stressed the relevance of linguistics in text processing because textual data are in natural language. However, the existence of paraphrase in language and the necessity within an information retrieval context for normalizing or standardizing the information content of such expressions has resulted as well in an emphasis on the importance of logic. Most applications of the functional calculus to data in natural language form have been unprofitable, because relatively few sentences can be characterized as logical propositions in any useful way. Non-standard logical systems, of the kind advanced by Williams and others (cf. Williams, 1965, 1966) to deal with thematic or topic representation, have made it possible to treat a broader range of natural language expressions.

Williams has been working for some years on the development of a "logistic" grammar. Within this grammar she provided a schema that, applied to natural language data, embodies many of the distinctions which seem central to semantics. The similarity of these distinctions to those from recent work in transformational linguistics prompted a careful comparison. The result was a demonstration of a systematic

correspondence between the underlying structures of a transformational grammar and the "semiographs" of a logistic grammar. In a practical implementation of this result, base trees of a grammar for a small section of English were converted mechanically into simplified predicate-argument structures. This convergence of linguistic research with research in applied logic provided the conceptual foundation for the design of the text-processing system by the project.

C. List-Processing Programming Languages

List-processing programming languages were developed in response to the requirements of programming in areas like computational linguistics, artificial intelligence, information retrieval, mechanical translation, and algebraic formula manipulation. The standard programming languages which have been developed for handling numerical data are not convenient for symbol manipulation. They are particularly inadequate for manipulating those complex symbolic structures in which information is carried by the relations among data items as well as by their content.

List-processing languages deal with data represented as list structures. A list is a sequence of elements. A list structure is a list whose elements may themselves be lists. The elements in a list are related to each other in computer memory by means of addresses that point to successive members. A whole list may be referred to by the address of the first element; insertions and deletions can be made by changing addresses. As a result, memory space within the computer does not have to be preassigned but can be allocated dynamically as required.

Dynamic storage allocation also makes it possible to reclaim and reassign memory space that is no longer necessary for a structure. Since the analysis of logical and linguistic data requires many intermediate steps, this feature is extremely useful.

List-processing languages are particularly well suited for performing recursive operations in which a subroutine may contain references to itself. To preserve the interim partial results prior to a recursive self-reference, pushdown stores have proved useful. A new element is added to a pushdown store on top of the existing elements as the first member of the list. Elements are retrieved on a last-in-first-out basis.

The features of list-processing languages that make them well suited for manipulating symbolic expressions also make them well suited for on-line programming. In fact, a program written in a list-processing language may itself be a list structure. On-line programming and the manipulation of complex linguistic structures can be accomplished particularly effectively by a language like LISP (McCarthy, et al., 1962) which was designed specifically to handle recursive functions in mathematics. However, the LISP notation for programming is somewhat awkward; the TREET language (Haines, 1965), an adaptation of LISP using a more natural notation, has been implemented within the MITRE computing facility (Haines, 1967). The preliminary model of the text-processing system is written in TREET. Furthermore, much of the design work for the model system was performed on-line using the TREET-based "On-Line Programming System" (Gross, 1967).

D. An On-Line Computer Facility

Interaction between a user and a computer is essential in a text-processing system, because the analysis necessary for representing in depth the information content of text cannot yet be wholly mechanized. In addition, although a computer can provide capacity and speed, the analyst is needed to monitor and guide its operations at critical points in the flow of processing. For these reasons, a text processing system must be implemented in an on-line computing facility.

The kind of on-line facility necessary is determined by two further considerations. First, the complex kind of analysis involved in processing text requires the central processing capability of a large computer. Second, if the system proves to be useful for an analyst, it would be desirable to make it available simultaneously to a number of analysts. These two considerations make time-sharing essential; however, extensive use of time-sharing requires some way to minimize the overhead costs entailed in handling multiple users. One technique that needs to be explored involves the concept of distributed processing. In distributed processing, computers of different sizes are linked together, and tasks are assigned to each in terms of the computer capacity required to accomplish them. As a consequence, the larger computers would be interrupted only for the more complex kinds of processing.

The particular equipment configuration required for the analyst is dictated by the type of data he deals with and by the desirability to limit the need for him to master complex programming skills. Display consoles are singularly appropriate for textual data because they provide visual

access to documents and messages stored in the computer memory. Lightgun controls and the use of "lightbuttons" -- commands initiated by lightgun actions on the face of the display scope -- have proved to be particularly useful devices for allowing a user to interact with displayed data. (See Section III.E. below for a description of the facility used in the implementation of the preliminary model.)

Display consoles and lightguns used in a facility with distributed processing can provide for efficient interaction, but only if there is appropriate programming support. The list-processing programming language described in the previous section provides the proper software control; moreover, it has the flexibility required so that changes in program structure can be made easily. Consequently, further results of the project work can continue to provide capabilities that can be added to the text-processing system.

III. Current Status

A. Introduction

The preliminary model of the text-processing system allows an information analyst to scan through messages, reports, and other documents and to select relevant facts, which are processed linguistically and then stored in the computer in the form of logical content representations. To satisfy a specific requirement the analyst can initiate a search through the stored representations to identify those with relevant content. Once identified, the original texts from which those representations were derived can be recovered, and, with suitable on-line editing and the addition of commentary and interpretation by the analyst, formed directly into reports on the topics under consideration.

This model, although indicative of the feasibility of the overall system design, needs to be augmented in order to form an experimental system that would be adequate for an analyst to test out in the context of his own problem area. A considerable amount of work has been done toward providing the additions that seem to be necessary. After a more detailed presentation of the model and a sketch of the overall system design, the current status of the proposed additional components will be described. The remainder of this section will contain descriptions of the computer facility and the programming language and system within which the model has been implemented.

B. Preliminary Model

The model system can conveniently be regarded as having two phases, representation and recovery. The input to the representation phase is raw text--a message, a news article, a report, etc. After it has been read

into the computer, the text is available for display on a CRT to the analyst, who reads in the text and extracts from it the salient facts which he wishes to store. He then phrases these facts in simple English declarative sentences, which are called "kernels", and submits them to the machine for processing and storage. A kernel may be constructed on the display by lightgunning, a word at a time, the component words from the displayed raw text, using the typewriter to introduce desired words not present in that text. Alternatively, the kernel itself may be typed in in its entirety. When a kernel is complete, the analyst lightguns the command "process", and programs for syntactic analysis and semantic interpretation convert the sentence automatically into a content representation. The analyst then can begin to construct another kernel.

If for any reason a kernel cannot be processed, the system provides the analyst both with information about the nature of the problem and with the basis for correcting it. The nature of this interaction between the analyst and the system can be appreciated best in the context of a more detailed description of the procedures involved.

The first step in the machine processing of a kernel is a lexical lookup of every word in the kernel. If any words not in the machine's current lexicon are encountered, the analyst is asked to classify them. At present, the classification is merely part-of-speech assignment as noun, verb, or adjective. The list of classes is displayed, and the appropriate one lightgunned. This information is internalized by the machine and retained for future reference. When it has classifications for all the words in a kernel, the machine attempts to parse it according

to the grammar of the system. If there are no parsings, it can be for one of two reasons. Either the kernel is simply not acceptable to the grammar, in which case it must be rephrased, or there is a wrong lexical classification in terms of the current kernel (for example, supply may be listed in the lexicon only as a verb, while in the current kernel it is being used as a noun). To ascertain whether the latter is the case, the machine displays all of the words in the kernel with their associated classifications. The analyst scans this list, and, if a classification is incomplete, he supplies the necessary additional information. The lexicon is updated and the kernel reprocessed.

If more than one parsing is obtained, the kernel is ambiguous and the analyst is informed of this fact so that he may rephrase.

The optimum and most usual case is that exactly one parsing is obtained. The machine makes a semantic interpretation of the kernel based on its grammatical structure. This interpretation at present consists of determining the activity expressed in the sentence; the subject, object, place, and time of the activity; and any property or indication of quality associated with the subject, object, and place. These facts are structured into a logical form which is stored as the content representation of the kernel.

The recovery phase begins with the analyst informing the machine that he wishes to make a query. The machine displays a list of categories corresponding to the kinds of information it possesses (those identified in the previous paragraph). The analyst enters in this display, via typewriter and lightgun, the particular things he wishes to find out more about,

with query indicators on the categories he desires the machine to fill in. For example, if he wishes to find out the actions taken by American planes during July, he enters by SUBJECT planes, by PROPERTY (of subject) American, by TIME July or during July, and by ACTION a Q (query). The machine searches its store of content representations of previously input kernel sentences. If any meet the specifications, they are searched in the queried categories. The information obtained is displayed, appropriately labeled, below the still-showing query list. (If none meet the specifications, the analyst is informed of this.) Confronted with the new information, the analyst may sharpen, broaden, or change his query as he likes, and the machine will recompute the information accordingly.

The analyst may at this point be satisfied with what he has learned. He may, on the other hand, want to see the context of some just-learned fact. By lightgunning any of the displayed answers, he can retrieve a display of the original section of raw text from which the kernel yielding that answer was derived. Editing operations can then be performed on this displayed text to reduce it to its essential elements, which may be stored with related portions of text. The accumulations of related text entries can themselves be displayed and edited; commentaries or interpretations can be inserted by the analyst; and then the result can be printed out on- or off-line for report generation.

Flow charts for the representation and recovery phases of the preliminary model are presented in Figure 1. A more detailed description of its operation is given in the SAFARI User's Manual (Chapin, et al., 1967). A film illustrating the system has also been prepared (Chapin, 1967).

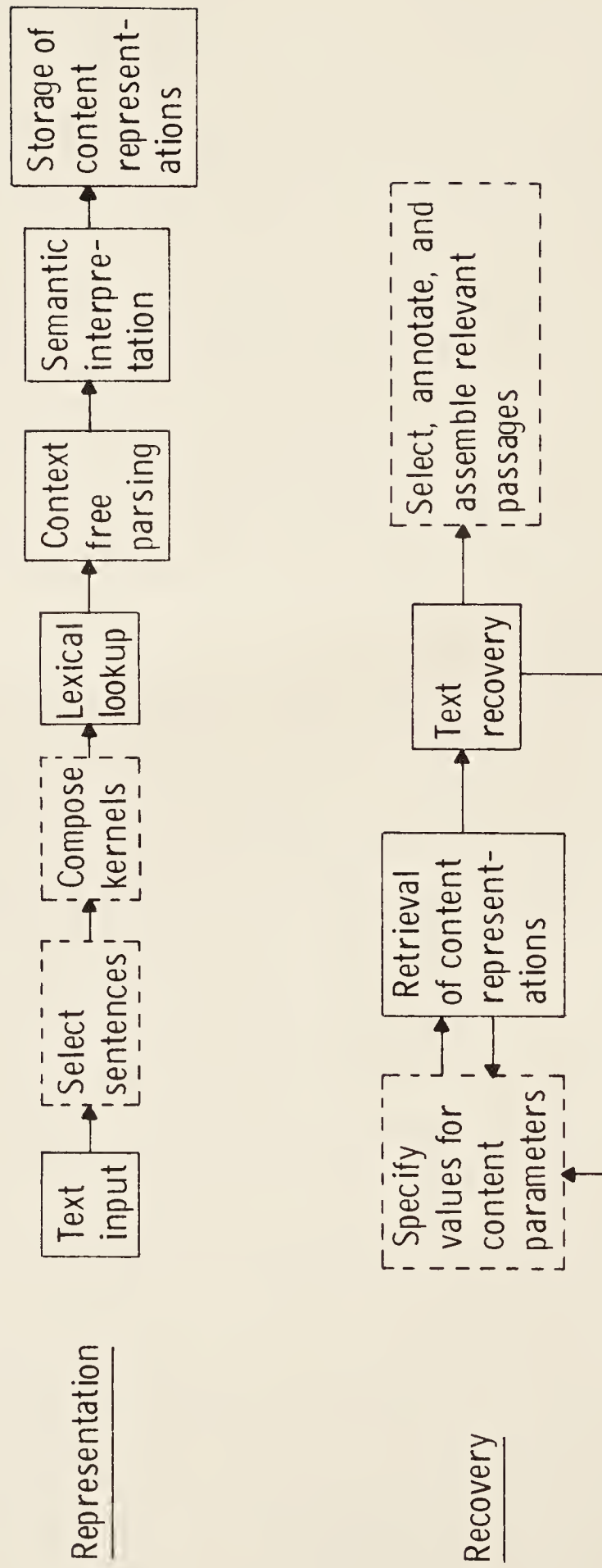


Figure 1. Flow charts for the preliminary model of a text-processing system. (Dashed lines indicate operator interaction.)

C. System Design

The proposed design for a text-processing system suitable for experimental testing by an information analyst differs in four major respects from that of the model, as well as in complexity and sophistication of processing. The major differences are: (1) the introduction of lexical analysis procedures to augment lexical lookup, (2) the replacement of context-free parsing of kernel sentences by a transformational analysis which can operate on more complex sentences, (3) the introduction of thesaurus and dictionary procedures to relate lexical items to each other for the retrieval of associated content representations and for expressing aspects of discourse structure, and (4) the development of techniques to group content representations for more efficient storage and retrieval.

In the current lexicon each variant form for a word must be listed; the introduction of morphological analysis procedures to identify stems and affixes would reduce the number of entries significantly. Further sophistication would be provided by adding to each lexical entry syntactic features which provide more complex information on its grammatical category assignments and which specify restrictions on its co-occurrence with other entries. The resulting lexical analysis procedures would improve the efficiency of the syntactic analysis and would allow content representations containing common stems to be automatically related to each other.

The replacement of context-free parsing by a transformational syntactic analysis would increase significantly the types of grammatical constructions covered by the system; it also would reduce the demands on the user during input processing of text. The rephrasing required of the

analyst would be reduced to editing out constructions not in the grammar. The transformational analysis automatically converts complex sentences into their component kernels and preserves the interrelations of these kernels to each other in the form of base trees.

The introduction of thesaurus and dictionary procedures would allow varying degrees and kinds of relatedness among words to be specified. Using such procedures during the representation phase, a content representation could be linked automatically to representations containing synonymous terms. Used during the recovery phase, these procedures would allow the analyst to establish bounds on the retrieval of related representations.

The development of techniques to group related content representations will be essential for efficient storage and search operations, particularly in a large data base. They would also be useful for reorganizing data to test hypotheses.

In addition to these major differences between the preliminary model and an experimental system, other changes would be desirable. More highly differentiated content representations would allow more discriminating retrieval operations; work in semantics is necessary to provide the basis for an increase in sophistication of this kind. Simplification of the user interface would allow the analyst to be more efficient. For example, it might prove desirable to formulate queries in English sentences in the recovery phase; they would be processed by the same techniques used in the representation phase to produce content representations. Other adjustments, not entirely predictable, would certainly be encountered

in integrating the components to form an experimental system.

Flow charts for the representation and recovery phases of the experimental system are presented in Figure 2.

D. Proposed Additional System Components

A significant amount of the work required to extend the preliminary model to an experimental system has already been accomplished. The task efforts required for certain of the other system components are well-defined. It is apparent, particularly for the components listed below, that the results of work done on one will be relevant for others. This interdependence illustrates the coherence of the basic design concept.

1. Transformational Analysis

The transformational syntactic analysis procedure developed at MITRE is an operational program that produces all the base trees with respect to a given transformational grammar for given input sentences. If the procedure produces more than one base tree, the sentence is ambiguous with respect to the grammar. The current grammar can handle sentences with the following kinds of constructions: simple declaratives of the actor-action (with or without direct object) and subject-predicate (predicate nominative, adjective, or locative) types, in any tense, with or without modals; coordinate compounds; passives; existential assertions ("there is" - "there are"); restrictive relatives; reduced relatives; preposed adjectives; yes-no questions; WH-questions; negatives; imperatives; maxima, minima, averages; nominalized measure adjectives; distance and duration complements; "type of" and "class of" constructions; adverbial modifiers of location, distance, time, rate, frequency, origin, destination; adnominal modifiers of

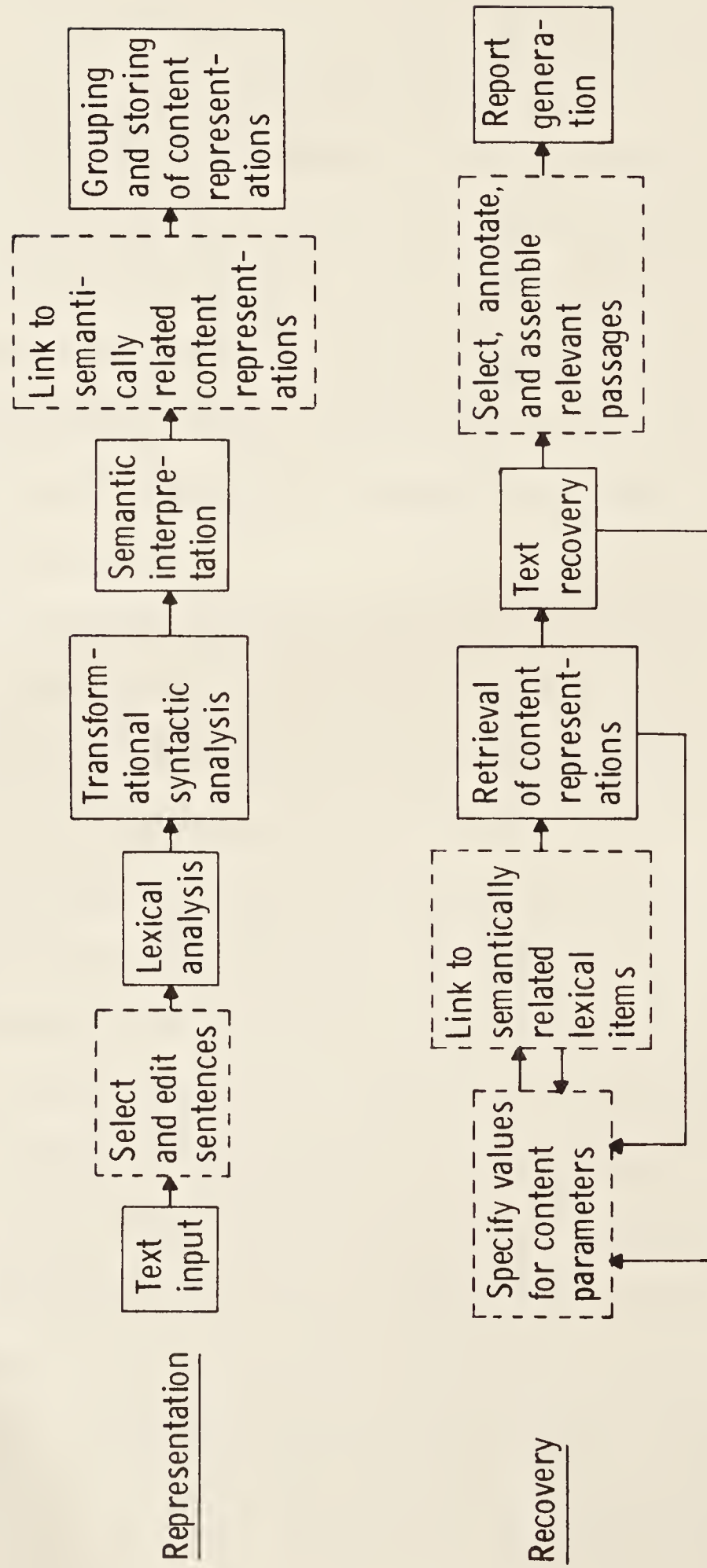


Figure 2. Flow charts for the experimental version of a text processing system. (Dashed lines indicate operator interaction.)

quantification, logical and numerical. Considerable work also has been done on comparatives and noun-noun compounds and on a more general treatment of prepositional phrases.

A major modification of this grammar based on the latest developments in transformational linguistics is under way. Constructions on the list in the preceding paragraph from simple declaratives through yes-no questions have already been incorporated into a new experimental grammar. This grammar will provide base trees that should facilitate the process of semantic translation. Part of the power of the grammar results from simplifying the phrase structure rules while making the lexicon more complex. Lexical items, instead of being treated as unanalyzable grammatical symbols, will have syntactic features, which provide information about subcategorization and selectional restriction and thus establish conditions on wellformedness to constrain the sentences in which they can occur.

2. Morphological Analysis

A set of procedures for morphological analysis motivated by transformational linguistics has been under development for some time. Rules for lexical derivation are being written which take into account the syntactic and semantic interrelations of stems and affixes considered as linguistic entities. These rules constitute an ordered cyclical set of strictly local transformations that operate on specified lexical items to produce derivations with specified characteristics. An experimental computer program, incorporating a small set of rules, has been written to test out the approach; preliminary results are encouraging.

3. Thesaurus and Dictionary Procedures

Relations among content representations need to be identified for both more effective storage and more adequate retrieval. An experimental investigation was undertaken to determine if the synonym groupings contained in published reference words could be incorporated into the text-processing system to retrieve such relations mechanically. The initial approach used a dictionary form Roget's Thesaurus, which under "major category" headings groups classes of synonyms and under individual entries refers to one or more of these major categories. Words were related to each other either by direct or by indirect linkages through major categories.

The results of work with the Roget were promising, although the sets of related terms tended to be too general and the resulting chains of relations became too inclusive. The introduction of hierarchical and other kinds of relations among the terms within a set would have been helpful. In general, the intervention of the analyst to constrain the synonym sets on-line seems desirable; the presence of the text itself would make appropriate discriminations easy to accomplish. Interactions between the analyst and the system would be essential in any case to relate proper names and terms used with special senses. Microthesauruses, synonym dictionaries, and thesauruses other than Roget's are also being examined because of the desirability of mechanizing as much of the task as possible. Whether the requirements of a particular application area will prompt the construction of a special thesaurus remains to be determined. Experiences with a specific data base will help resolve this question.

In the course of the thesaurus work, its relevance to the interpretation of sentential meaning, to paraphrase, and to the resolution of structural ambiguity within discourse has become increasingly evident. This approach to the determination and representation of discourse structure is being pursued.

4. Semantic Analysis

A considerable amount of work has been done on the semantic component of a transformational grammar in an attempt to provide a basis for determining the meanings of phrases from the meanings of the lexical entries which compose them. Katz and his colleagues at M.I.T. (Katz and Fodor, 1963; Katz and Postal, 1964; Katz, 1966) proposed that a semantic interpretation for a sentence resulted from applying special "projection" rules to dictionary entries that contained semantic markers identifying the concepts (like "physical object", "human", or "male") of which words were composed. A projection rule related the meanings of the words in a syntactic construction in terms of the grammatical relation holding among them.

In the original formulation, the semantic markers for a word bore a uniform relation to that word. For the project, however, it has proved desirable to consider "second-order" markers that act as predicates and take markers and syntactically structured markers as arguments. The second-order marker makes it possible to specify different kinds of relations between the dictionary entry and the markers that characterize it. One of these relations corresponds to the "core meaning" of a word; this concept has proved to be particularly useful in the treatment of metaphor.

5. Discourse Analysis

An examination of selected newspaper articles and abstracts has prompted studies of certain aspects of discourse structure. The initial focus was on determining the degree to which raw text could be accommodated by the then current programs for processing language data; the results led to the introduction of the kernelizing procedure in the system. An equally important concern was the establishment of an experimental data base that contained paraphrases--in this case, the treatment of the same news event by different wire services and different editorial staffs. A data base so constituted would be useful for evaluating the adequacy of any system for content representation.

Studies of actual texts also led to a concern for problems of anaphora and for ways to deal with other kinds of intersentential reference. An algorithm for locating the antecedents of pronouns was developed; it seems to work in a variety of cases and merits further study. Related work involving the use of thesaurus relations also seems promising.

E. Computer Facility

The preliminary model of the text-processing system has been programmed on the IBM 7030 (STRETCH) computer. The 7030 has a core memory containing 65K 64-bit words; a model 353 double-density disk storage unit holds four million words. There are five user stations, each containing a Data-Display-13 display console (the NORAD prototype) with photoelectric lightgun, a modified IBM Selectric on-line typewriter, and a Stromberg-Carlson 3070 medium-speed printer.

The IBM Master Control Program for the 7030 has been rewritten to allow for time-sharing the operation of the display consoles as a foreground job with a batch-processing background mode. The five display consoles operate independently within the TREET system; each console user works with his own core load; swap time (core to disk and disk to core) is about 1/2 second.

F. The TREET Programming System

TREET is a general purpose list-processing programming system that has been implemented for on-line and off-line use on the IBM 7030 computer. The word TREET is used to mean the following things: (1) a specific language (the TREET language), (2) an evolving abstract entity consisting of a group of languages, including the TREET language, and a set of definitions and conventions together with a philosophy relating to data structure, program structure, etc., and (3) a collection of programs for the 7030 (cf. Haines, 1967, for a detailed description).

The TREET language (cf. Haines, 1965, for a detailed description) is a list-processing language that has the capabilities of LISP 1.5, but looks more like ALGOL or FORTRAN to the user. TREET operates on list structures, and all data must be coded in that form. Within TREET, a list structure is an atom or a list of list structures. An atom is a symbol (of from 1 to 10 alphanumeric characters), a number (integer or floating point), or a card (an internal image of an 80-character IBM card). A list may have any number of members, and each member may be arbitrarily complex; a list is written by enclosing its members within parentheses.

All programs in the TREET language are written as functions. A function normally has a single value (which may be an arbitrarily complex list structure), a unique name, and operates with zero or more arguments. A function may or may not have an effect on the system other than its value; some functions are used only for their effect, others only for their value, others for both effect and value.

The other languages that with the TREET language proper can be used to code functions accepted by the TREET system in the current implementation include Cambridge Polish, TAP, and SMAC. Cambridge Polish is a parenthesized prefix language, used in LISP, that is machine independent, but particularly easy for the computer to process. TAP, the TREET Assembly Program, is a basic macro-language whose syntax is machine independent. SMAC is the STRETCH (IBM 7030) macro-language, a machine-dependent language that serves as the lowest level programming language. Functions may be defined on-line in TREET, Cambridge Polish, and TAP.

The philosophy underlying TREET can be characterized in several statements. The TREET language was designed to be convenient to use, easy to learn, highly intelligible to anyone familiar with computer programming, efficient in its ability to express algorithms, efficient in operation, and easily implemented in a computer. The implementation has stressed machine independence with maximum communication among the languages used in the system and maximum access by the programmer to each of those languages.

The collection of programs that forms the TREET system for the IBM 7030 consists of a supervisor, a translator to relate the languages to

each other, and a set of subroutines that provide a basic programming capability. The supervisor controls the overall operation of the system. The system translator includes a translator that converts TREET to Cambridge Polish, an interpreter that "executes" the Cambridge Polish, a compiler that converts Cambridge Polish to TAP, and a macro assembler that converts TAP to machine language. The basic system subroutines include functions that can be used on-line or off-line for list-processing, for performing arithmetic operations, for manipulating data in card format, for directing input/output operations, for performing standard programming operations like conditional, branch, and return, and for doing program diagnostics and debugging.

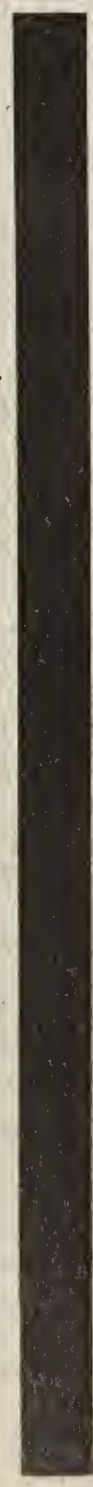
G. Program Structure for the Text-Processing System

The preliminary model of the text-processing system was programmed in TREET. TREET also was the vehicle for its development, with extensive use being made of the on-line programming system (Gross, 1967) within the display facility. The program structure reflects the flow charts for the system provided in Figure 1.

The functions for selecting sentences, composing kernels, specifying content parameters, and selecting and assembling relevant passages require operator interaction and thus are specific to the facility configuration and, to that extent, machine dependent. However, the functions for lexical lookup, context-free parsing, semantic interpretation, storage of content representations, retrieval of content representations, and text recovery are to a considerable extent machine independent. All of the functions are relatively self-contained; they are linked to form a modular operating system by calling each other in a relatively simple fashion.

References

- Chapin, P. G. SAFARI: a progress report on the development of a text-processing system. Film, MITRE Corporation, 1966.
- Chapin, P. G., Gross, L. N., Norton, L. M., Beller, R. J., and Browne, C. T. SAFARI, an on-line text-processing system: user's manual. MTP-60, MITRE Corporation, 1967.
- Chomsky, N. Aspects of the theory of syntax. Cambridge, Mass.: M.I.T. Press, 1965.
- Gross, L. N. On-line programming system: user's manual. MTP-59, MITRE Corporation, 1967.
- Haines, E. C. The TREET list processing language. SR-133, MITRE Corporation 1965.
- Haines, E. C. The TREET programming system: IBM 7030 implementation. MTP-58, MITRE Corporation, 1967.
- Katz, J. J. The philosophy of language. New York: Harper & Row, 1966.
- Katz, J. J., and Fodor, J. A. Structure of a semantic theory. Language, 1963, 39, 170-210.
- Katz, J. J., and Postal, P. M. An integrated theory of linguistic descriptions. Cambridge, Mass.: M.I.T. Press, 1964.
- McCarthy, J., et al. LISP 1.5 programmer's manual. Cambridge, Mass.: M.I.T. Press, 1962.
- Walker, D. E. (Ed.) English preprocessor manual. SR-132, MITRE Corporation, 1965.
- Walker, D. E., Chapin, P. G., Geis, M. L., Gross, L. N. Recent developments in the MITRE analysis procedure. MTP-11, MITRE Corporation, 1966.
- Williams, T. M. Some logical aspects of thematic representation with explicit ties. Paper presented at the 1965 Congress, International Federation for Documentation, Washington, D. C.
- Williams, T. M. Case variables and case description in a reticular logistic grammar. Unpublished manuscript, 1966.
- Zwicky, A. M., Friedman, J., Hall, B. C., Walker, D. E. The MITRE syntactic analysis procedure for transformational grammars. AFIPS Conference Proceedings: Fall Joint Computer Conference, 1965, 27, 317-326.



THE
MITRE
CORPORATION